Lecture - 1,2, Introduction to Computer Vision and Image Classification

Sai Amrit Patnaik

July 2020

These notes are my personal notes to the course CS231n Convolutional Neural Networks for Visual Recognition offered by Stanford university. Notes are from the 2016 winter offering of the course taught by Andrej Karpathy and Justin Johnson. The Introduction is given by Professor Fei Fei Li.

1 Introduction

- The Primary Visual Cortex is the region where the brain does most of the vision related tasks.
- Almost 50% of the brain is used in Visual Processing. Vision is the hardest and most important sensory perceptual cognitive system of the brain.
- **Two important Insights of Visual System :**
  - Neurons in the Primary Visual Cortex are arranged in Columns. **Each column is excited by a particular type of stimuli (Eg: Horizontal lines, vertical lines etc.)**
  - The second most important insight about vision is that, **Vision is hierarchical**. It starts by recognizing blobs, edges etc simple features in the initial phases and gradually learns complex features in the later stages.
- Human Visual System is an ill posed problem, i.e-real world is in 3D but the image that is created on the retina is a 2D image.
- **Visual Perceptual Grouping :** Perceptual Grouping says that when we see an image, we don't see the bigger picture at a time, we segment and group things and apply understanding over the groups.

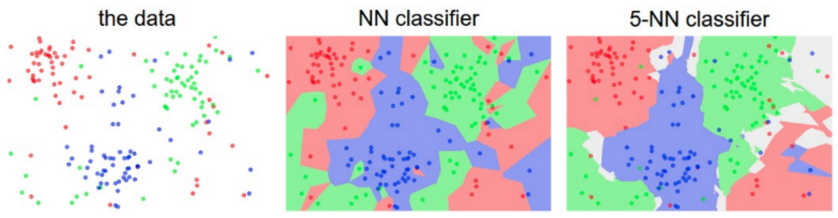
2 Image Classification

- Image classification generally takes a data driven approach to solving the problem. In data driven approach, a lot of training samples of each class are shown to the system and the machine learns an understanding about the visual appearance of each class.The classification pipeline is as follows:

- Develop a training set of images
- Train the learning algorithm on the training set of images.
- Test the learned model on a test set which model has never seen.

3 K-Nearest Neighbour Classifier (KNN)

- The learning algorithm takes all the training data and remembers all the data and labels.
- At test time, compares the test point with all the training points and assigns a label to the test point.
- In K nearest neighbours, the label of the test point is decided by taking a majority voting of the labels of the k nearest data points.



- Commonly used distance metrics:
  - **Manhatten Loss(L_1) :**

$$d(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

  - **Euclidean Distance(L_2) :**

$$d(I_1, I_2) = \sqrt{\sum_p |I_1^p - I_2^p|^2}$$

- Nearest Neighbour algorithm is linearly slow in test time with training data, the more the training data, the longer is the inference time.
- There are lot of hyper-parameters in the algorithm like the K value, distance metric etc, so it is best to test on a validation set separately prepared from the training set and set all hyper-parameters on the validation set and finally test the performance on the test set.
- If the amount of training data is too less, k-fold cross validation can be used.
- Because KNN is considerably slow at test time with increasing in training data and the general requirement is quite opposite, i.e - okay to be slow in training but fast at test time, so KNN is not a popularly used algorithm.

4 Linear classifier

- In the parametric approach, we try to develop a function that maps from pixels to labels which is a f^n of both inputs and a parameter which we aim to learn in the training process.
- The approach will have two major components:
  - A **Score function** that maps the raw data to class scores
  - A **Loss function** that quantifies the agreement between the predicted scores and the ground truth labels.
- The simplest model would be a Linear function.

  $$f(x_i, W, b) = W \times x_i + b$$

  The matrix W and B are of shape $(D \times k)$ and $(1 \times k)$ where k is the no. of classes. So W and b are the parameters.
- We will then cast this as an optimization problem in which we will minimize the loss function with respect to the parameters of the score function.
- **Features of Parametric Approach :**
  - Each row of the weight matrix corresponds to one classifier per class.
  - The k^{th} row of the classifier corresponds to how much weight is being given to each pixel to classify the input image as class k.

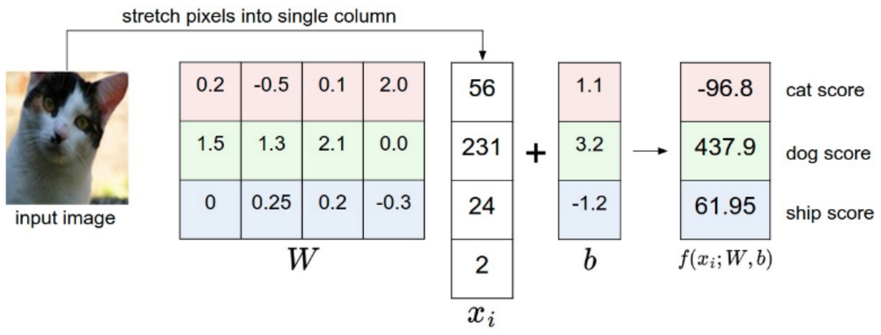
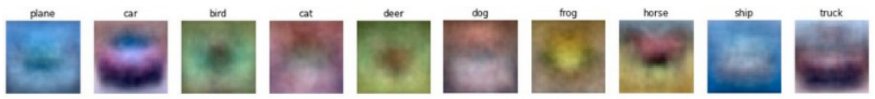


Figure 1: In the figure above, the 1st row in the W matrix correspond to the cat classifier and the value in the row correspond to the weights it gives to each pixel of the image to make a prediction if it is a cat or not.

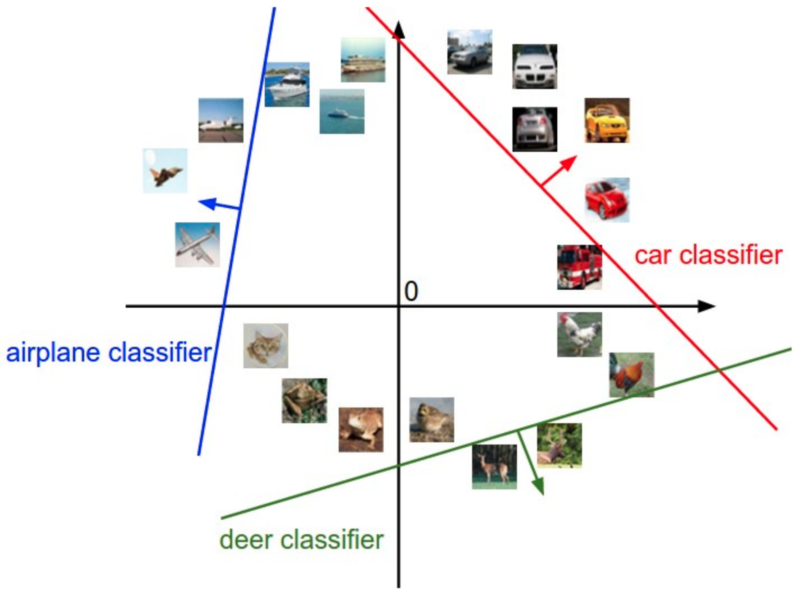
  - Weight matrix is k individual classifiers stacked together.
  - Each row of the W matrix is a template image for the particular class that the model learns from the input data.



Skipping ahead a bit: Example learned weights at the end of learning for CIFAR-10. Note that, for example, the ship template contains a lot of blue pixels as expected. This template will therefore give a high score once it is matched against images of ships on the ocean with an inner product.

  - Now to find its similarity with an input image, Dot product is the best way to find similarity (dot product = 0 being exactly similar and dot product = 90 means it is orthogonal)

- Once the training data is used to learn the parameters, the entire training data can be totally discarded and only the learned parameters can be used for inference.
- Each input image is a point in the image space where each row of the W matrix corresponds to a hyper plane in the image space separating the points from each other into different classes classifying the region into classes.



Cartoon representation of the image space, where each image is a single point, and three classifiers are visualized. Using the example of the car classifier (in red), the red line shows all points in the space that get a score of zero for the car class. The red arrow shows the direction of increase, so all points to the right of the red line have positive (and linearly increasing) scores, and all points to the left have a negative (and linearly decreasing) scores.

- Every row of W is a classifier for one of the classes. The geometric interpretation of these numbers is that as we change one of the rows of W, the corresponding line in the pixel space will rotate in different directions.
- The biases b, on the other hand, allow our classifiers to translate the lines. In particular, without the bias terms, plugging in $x_i = 0$ would always give score of zero regardless of the weights, so all lines would be forced to cross the origin.

5 Loss Functions

To quantify the amount of error caused by the chosen parameters, loss functions are used.

5.1 Multi Class Support Vector Machine (SVM) Loss

- This loss is setup such that the scores of the correct class are separated from the incorrect classes by a margin Δ.
- For an image suppose x_i is the image and y_i is the label corresponding to it. The score for an input example j is $S_j = f(x_j, W)$, the SVM loss for an example x_i is given as :

$$L_i = \sum_{j \neq y_i} max(0, S_j - S_{y_i} + \Delta)$$

- Total Loss :

$$L = \frac{1}{N} \sum_i L_i$$

- For incorrect classes that get **a score Δ less than the score of the correct class, get a 0 loss** and incorrect classes having **score more than correct class, or are lesser than a margin Δ get positive loss.**
- The problem with SVM loss is that the W values that satisfy the loss are not unique. Suppose a matrix W satisfies the loss, any scaled value of W would also satisfy because this only is concerned with the difference of scores which is invariant of scale. Solution discussed in section 6

5.2 Soft Max Loss

- The score $f(x, W) = W \times x$ is interpreted as un-normalized probabilities, so we take exponent of the scores and then normalise the exponentiated scores. The loss is defined as :

$$L_i = -\log_e \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right)$$

5.3 Interpretations of Soft-max

- **Information theory View** :
  - Cross Entropy between true distribution p and estimated distribution q is defined as :

$$H(p, q) = -\sum_x p(x)log(q(x))$$

  - So softmax is trying to minimize the cross entropy between estimated class probabilities, $q = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$ and true distribution where entire probability mass is on the same class([0,0,..,1,..,0,0] all zeros, 1 at only the $y_i^{th} position$)
  - Moreover, since the cross-entropy can be written in terms of entropy and the Kullback-Leibler divergence(KL Divergence) as : $H(p, q) = H(p) + D_{KL}(p||q)$ and the entropy of the delta function p is zero.
  - This is also equivalent to minimizing the KL divergence between the two distributions (a measure of distance). In other words, the cross-entropy objective wants the predicted distribution to have all of its mass on the correct answer.
- **Probabilistic View** :
  - The expression $P(y_i|x_i; W) = -\log\left(\frac{e^f_{y_i}}{\sum_j e^f_j}\right)$ can be interpreted as minimizing the negative log of normalized probability assigned to correct class.
  - We are therefore minimizing the negative log likelihood of the correct class, which can be interpreted as performing Maximum Likelihood Estimation (MLE).
  - We can now also interpret the regularization term R(W) in the full loss function as coming from a Gaussian prior over the weight matrix W, where instead of MLE we are performing the Maximum a posteriori (MAP) estimation

5.4 Handling Instabilities of Soft-max

- For computing the Softmax function in practice, the intermediate terms $e^{f_{y_i}}$ and $\sum_j e^{f_j}$ may be very large due to the exponentials.
- So we multiply the top and bottom of the fraction by a constant C and push it into the sum, we get the following (mathematically equivalent) expression:

$$\left(\frac{e^f_{y_i}}{\sum_j e^f_j}\right) = \left(\frac{C \times e^f_{y_i}}{C \times \sum_j e^f_j}\right) = \left(\frac{e^{f_{y_i}} + logC}{\sum_j e^{f_j} + logC}\right)$$

- C can be any value, that won't change any value but would just increase numerical stability of computation and save from exponent to explode.
- A common choice for C is to set log$C = -max_j f_j$. This simply states that we should shift the values inside the vector f so that the highest value is zero.

6 SVM loss VS Softmax loss

- Performance of both is almost similar.
- One very important thing between both is :
  - SVM is satisfied with the Δ difference between correct class scores and incorrect class scores. So ([10, 9, 9] and [10, -100, -100]) have same effect from the loss.
  - For the same scores, Softmax imposes a high loss on [10, 9, 9] and lower loss to [10,-100, -100] because of probability.
  - The Softmax classifier is never fully happy with the scores it produces.
  - The correct class could always have a higher probability and the incorrect classes always a lower probability and the loss would always get better.

7 Regularization

- To remove the ambiguity of multiple W values giving same loss and to find a preferred value of W, **Regularisation** is used. The regularizing term is added to the loss. The most commonly used Regularizing terms are :
  - L_2 Regularization, $R(W) = \sum_i \sum_j W_{i,j}^2$
  - L_1 Regularization, $R(W) = \sum_i \sum_j W_{i,j}$
- Total Loss : $L = \frac{1}{N} \sum_i L_i + \lambda R(W)$
- L_2 regularization prefers the weights values to be more distributed and lower values.(Eg. [0.25, 0.25, 0.25, 0.25] preferred over [1, 0, 0, 0])
- L_1 regularization prefers more sparse weights where lots of weights are zeroed down. So L_1 is mostly used for feature selection bu zeroing down dimensions.